Amendments to the Claims

This listing of claims will replace all prior versions and listings of claims in the application:

Listing of Claims

1. (Original) A method for creating a plurality of objects from data in persistent storage, the objects having object pointers, unique object identifiers, and object types as attributes, the method comprising the steps of:

reading the total number of type sets;

for each type set, reading the total number of objects in each type set;

for each object in said type set, creating an object from said data in persistent storage;

for each object pointer in said objects, obtaining the unique object identifier corresponding to said object pointer; and

for each obtained unique object identifier, obtaining the object address corresponding to said unique object identifier and setting each of said object pointers to said corresponding object addresses.

2. (Original) A method as recited in claim 1, wherein said objects are created in a first pass and said objects' pointers values are set in a second pass subsequent to said first pass.

3. (Original) A method as recited in claim 1, wherein for each of said object pointers in each of said objects:

attempting to obtain said object addresses, and setting said object pointers equal to said object address if said pointed to objects exist, otherwise deferring said setting object pointer step until said object exists.

4. (Previously Presented) A method for writing a plurality of objects in non-persistent storage to persistent storage, the objects having pointers to objects, unique object identifiers, and object types as attributes, the method comprising the steps of:

providing one or more common interfaces that are used by each of the plurality of objects to write the objects from non-persistent storage to persistent storage;

grouping together said objects into type sets, wherein each of said objects in each of said type sets have the same type, wherein each of said type sets have a set population equal to a total number of objects inhabiting said type set;

counting each of said type sets and arriving at a total number of sets;

converting each of said objects to a persistable form including obtaining a persistable form for each of said pointers to objects by obtaining a unique object identifier corresponding to each of said pointers to objects;

writing said total number of type sets to persistent storage; and

writing each of said type sets to persistent storage.

5. (Previously Presented) A method for storing and restoring user objects to persistent storage, the method comprising the steps of:

providing one or more common interfaces that are used by the user objects to store and restore the objects to/from persistent storage;

creating a persistence controller object for managing the persistence of the user objects, the persistence controller object being derived from at least one of the common interfaces;

providing a plurality of user defined classes, the classes derived from a common object base class;

creating a plurality of instances of user objects belonging to the user defined classes;

providing a stream-in method and a stream-out method for each of the user defined classes;

registering each added user defined class and added user object in a registry;

grouping the objects according to class;

storing the grouped user objects to persistent storage using the stream-out methods;

loading the stored objects from storage into memory using the stream-in methods; and

registering the user objects in the registry.

6. (Original) A method as in claim 5, wherein at least some of the user defined classes have pointers pointing at objects derived from the base class, wherein, when the pointers have a unique identifier associated with the pointer, wherein the saving step includes saving the unique identifiers associated with the pointers, and wherein the loading step includes setting the unique identifier value for each loaded object, obtaining the new address of each loaded user

object, and using the stored unique identifier associated with each pointer along with the new address to set each pointer value to the new address.

7. (Currently Amended) A computer readable medium encoded with an ~~An~~ object ~~adapted~~ for persistent storage, the object having a smart pointer, wherein the smart pointer includes an address attribute for containing [[the]] an address of [[an]] a subject object being pointed to by the smart pointer, and an object unique identifier attribute for containing [[the]] a unique identifier [[of an]] for the subject object being pointed to, wherein the object smart pointer has an assignment operation which ~~stores~~ correlates the address of the subject object being pointed to and the unique identifier of the subject object being pointed to, and wherein the object includes a load method for using the smart pointer unique identifier attribute to determine and load a new smart pointer address attribute for the subject object being pointed to after the subject object ~~being pointed to~~ is loaded from persistent storage.

8. (Currently Amended) An object as in claim 7, wherein the object includes a stream-out method for streaming out the smart pointer address attribute and/or the unique identifier attribute to persistent storage.

9. (Cancel)

10. (Previously Presented) A method for writing computer objects to persistent storage, the method including the steps of:

providing a plurality of software objects to be stored, wherein the objects are instantiations of at least one class to be storable, wherein the storage is in persistent storage, wherein each of the classes has a unique class ID, and wherein each of the objects has a unique object ID;

providing smart pointers for at least some objects, wherein the smart pointers include an address portion to contain the address of the object being pointed to and an object identifier portion to contain an object identifier of the object being pointed to;

providing a persistent object controller for controlling the lifecycle of objects to be saved to persistent storage and loaded from persistent storage;

providing a Persistent Object Registry for maintaining a database of objects to be saved to persistent storage, wherein the Persistent Object Registry is in communication with the persistent controller object;

providing a first save method to save all objects in the Persistent Object Registry to persistent storage;

providing a second save method for saving the attributes of each class having objects to be saved to persistent storage, wherein the second save method is called by the first save method;

providing a first load method for loading all objects saved in a file in persistent storage;

providing a second load method for loading the attributes of each class having objects to be loaded from persistent storage, wherein the second load method is called by the first load method;

registering the objects to be saved with the Persistent Object Registry using the persistent object controller, including storing the class ID and object ID of the objects to be saved;

writing the objects to be saved to persistent storage using the first save method and second save method;

reading the objects stored from persistent storage using the first load method and second load method;

registering the objects loaded into the Persistent Object Registry; and

resolving the smart pointer object address attributes by using the object ID attribute value to search the Persistent Object Registry to retrieve the current address of the object being pointed to.

11. (Original) A method as in claim 10, wherein the objects are all Component Object Model (COM) objects.

12. (Original) A method for managing persistent object lifecycles, the method comprising the steps of:

providing for each object a unique object identifier attribute, an object type attribute, and an object address;

providing an object registry object for maintaining a correspondence between said unique object identifier attribute, said object address, and said object type attributes;

creating a first object having a first object type, a first object address, and a first unique object identifier, and storing said first unique object identifier, address, and type in said object registry;

creating a second object having a second object type, a second object address, and a second unique object identifier, and storing said second unique object identifier, address, and type in said object registry, said second object having a pointer attribute set equal to said first object address;

providing said second object pointer attribute to said object registry and obtaining said first object unique identifier corresponding to said second object pointer attribute in return;

writing said second object to persistent storage as second object data, and writing said first object unique identifier corresponding to said second object pointer attribute to persistent storage, such that said written first object unique identifier is associated with said second object pointer attribute in persistent storage;

deleting said second object from non-persistent storage;

reading said second object data from persistent storage and creating said second object having said second object type;

reading said first object unique identifier associated with said second object data from persistent storage;

providing said object registry with said first object unique identifier and obtaining said first object address in return; and

setting said second object pointer attribute equal to said first object address, such that said second object pointer attribute again points to said first object.

13. (Previously Presented) A method for writing a plurality of objects in non-persistent storage to persistent storage, the method comprising the steps of:

providing one or more common object interfaces that are used by each of the plurality of objects to write the objects from non-persistent storage to persistent storage;

each of the common object interfaces having a corresponding common object class;

providing a Persistent Object Registry for maintaining a listing of objects to be saved to persistent storage, wherein the Persistent Object Registry is in communication with a persistent controller object, and the persistent controller object is derived from one or more of the common object classes; and

providing a first save method to save all objects in the Persistent Object Registry to persistent storage.

14. (Previously Presented) The method of claim 13 wherein each of the plurality of objects are derived from one or more of the common object classes.

15. (Previously Presented) The method of claim 13 further comprising the step of assigning a unique object ID to at least some of the objects.

16. (Previously Presented) The method of claim 15 wherein at least some of the objects have pointers to other objects, the method further comprising the step of identifying and recording the unique object ID of the other objects referenced by the pointers.

17. (Previously Presented) The method of claim 16 further comprising the step of saving the recorded unique object IDs of the other objects referenced by the pointers to persistent storage.

18. (Previously Presented) The method of claim 13 wherein the common object interfaces are COM interfaces including an IPersistFile interface, an IPersistStream and an IUnknown interface.

19. (Previously Presented) A method for reading a plurality of objects from persistent storage to non-persistent storage, the method comprising the steps of:

providing one or more common object interfaces that are used by each of the plurality of objects to load the objects from persistent storage to non-persistent storage;

each of the common object interfaces having a corresponding common object class;

providing a Persistent Object Registry for maintaining a listing of objects that are loaded from persistent storage, wherein the Persistent Object Registry is in communication with a persistent controller object, and the persistent controller object is derived from one or more of the common object classes; and

providing a load method to load objects in the Persistent Object Registry to non-persistent storage.

20. (Previously Presented) The method of claim 19 wherein each of the plurality of objects are derived from one or more of the common object classes.

21. (Previously Presented) The method of claim 19 further comprising the step of assigning a unique object ID to at least some of the objects.

22. (Previously Presented) The method of claim 21 wherein at least some of the objects have pointers to other objects, the method further comprising the step of identifying and recording the unique object ID of the other objects referenced by the pointers.

23. (Previously Presented) The method of claim 22 further comprising the step of loading the recorded unique object IDs of the other objects referenced by the pointers from persistent storage.

24. (Previously Presented) The method of claim 19 wherein the common object interfaces are COM interfaces including an IPersistFile interface, an IPersistStream and an IUnknown interface.

25. (Previously Presented) A method for storing and restoring user objects to persistent storage, the method comprising the steps of:

providing one or more Component Object Model (COM) interfaces that are used by the user objects to store and restore the objects to/from persistent storage, the one or more COM interfaces including a Persistent Object Interface derived from the IPersistStream class, and a Persistent Controller Interface derived from the IPersistFile class of the Component Object Model (COM);

creating a COM Persistence Controller object for managing the persistence of the user objects, the COM Persistence Controller object being derived from the Persistent Controller Interface;

providing a plurality of user defined classes, the classes derived either directly or indirectly from the Persistent Object Interface;

creating a plurality of instances of user objects belonging to the user defined classes; and

providing a stream-in method and a stream-out method for each of the user defined classes.